A Reasoning Methodology for CW-Based Question Answering Systems

Elham S. Khorasani, Shahram Rahimi, and Bidyut Gupta

Computer Science Department,
Southern Illinois University, Carbondale, IL, USA
{elhams,rahimi,gupta}@cs.siu.edu

Abstract. Question Answering Systems or (QA systems for short) are regarded as the next generation of the current search engines. Instead of returning a list of relevant documents, QA systems find the direct answer to the query posed in natural language. The key difficulty in designing such systems is to perform reasoning on natural language knowledgebase. The theory of Computing with Words (CW), proposed by Zadeh, offers a mathematical tool to formally represent and reason with perceptive information. CW views a proposition in natural language as imposing a soft/hard constraint on an attribute and represents it in form of a *generalized constraint*. In this paper we develop a reasoning methodology for the restricted domain CW-based QA systems. This methodology takes, as input, the knowledgebase and the query in form of generalized constraints and organizes the knowledge related to the query in a new tree structure, referred to as a *constraint propagation tree*. The constraint propagation tree generates a plan to find the most relevant answer to the query and allows improving the answer by establishing an information-seeking dialog with user.

Keywords: Reasoning methodology, Question Answering, Computing with Words, Generalized Constraint.

1 Introduction

The current search engine technologies are much limited to pattern matching and are still relied on human effort for providing useful information. Instead of a direct answer to the query, users receive thousands of documents that contain the input keywords and they have to manually process these documents to extract the desired information. The QA systems are regarded as the next generation of the current search engines. They receive a query expressed in natural language, process their knowledge base (KB), which is also in natural language, and return the most relevant answer to the query. Therefore QA systems need more complex natural language processing than other type of information retrieval systems. The key difficulty in designing such systems lies in the imprecise nature of natural language expressions. The theory of Computing with Words [7], which is rooted in fuzzy set and fuzzy logic, provides a mathematical tool to model the imprecision of natural language propositions and perform reasoning among perceptions. CW views a proposition in natural language as imposing a soft/hard constraint on an attribute and represents it in form of a Generalized Constraint (GC).GC provides a basis for approximate reasoning. It serves as a generalized language for representing different kinds of uncertainty such as probability, possibility, truth qualification and so forth. A proposition in natural language may be expressed in GC with the form GC : X *isr* R, where X is the constraint variable, R is the constraint on the values that X can take and is called the constraining relation, and r is the semantic modality of the constraint that specifies how R is related to X. There are three primary modalities which represent the three primary aspects of uncertainty: probabilistic (r=p), possibilistic (r=*blank*), and veristic (r=v). Other types of constraints can be viewed as the mixture of the primary constraints [7].

For example the proposition: "gas is expensive" can be represented in GC as: "*price(gas) is expensive*". New GCs may be derived from sets of existing GCs by conjunction, projection and propagation operations (For more details on generalized constraint theory refer to [7]).

After representing knowledge in form of a GC, a set of deduction rules need to be defined to perform reasoning. To do so, a GC is then summarized and abstracted into a protoform (PF), abbreviation for prototypical form. Informally a protoform is an abstracted summary of an object and represents the semantic of such objects [6]. For example the GC expression "*price(gas) is expensive*", can be abstracted to protoform:"*A(B) is C*", where A is an abstraction of linguistic variable "*price*", B is an abstraction of "*gas*" and C is an abstraction of the granule value "*young*". The concept of protoform plays an important role in reasoning; it allows classifying the objects of the same semantic structure and defining inference rules for manipulating them. These rules are drawn from various domains such as probability, possibility, fuzzy arithmetic, fuzzy logic and so forth and they basically govern propagation of GCs. Some examples of these rules are listed in table 1. More rules can be found in [7]. Each rule has a symbolic part, which is in terms of protoforms, and a computational part which defines the computation that has to be carried out to arrive at a conclusion. The focus of this paper is to develop a methodology that uses GC propagation rules to make a sequence of inferences on a GC knowledgebase, in order to provide an answer to the input query. Although fuzzy set theory and fuzzy logic are well defined and have been extensively studied in literature, there are not yet many works that extended and utilized CW to develop a working QA system.

Sufyan Beg et.al. [4] designed a hybrid framework for a QA deduction engine that combines the phrase-based deduction with CW reasoning. This framework identifies and tags the query as well as the sentences in KB and extracts the facts that are relevant to the query. If the relevant facts are tagged as a protoform, then they will be processed according to protoform deduction rules. Otherwise the standard bivalent logic reasoning will be applied to find the appropriate answer to the question.Ahmad, et.al. [1] proposed a framework for developing a CW-based fuzzy expert system for automated question answering. The focus of

Table 1. Examples of protoform deduction rules

rule	**symbolic part**	**computational part**
rule (1) interpolation	$\begin{array}{l} X\ is\ A \\ \sum_i if\ x_i\ is\ A_i\ then\ Y\ is\ B_i \\ \hline Y\ is\ B \end{array}$	$\mu_B(v) = \sum_i m_i \wedge B_i$ $m_i = sup_u(\mu_A(u) \wedge \mu_{A_i}(u))$, $i = 1, \ldots, n$
rule (2) intersection syllogism	$\begin{array}{l} Q_1 A's\ are\ B's \\ Q_2 (A\&B)'s\ are\ C's \\ \hline Q_2 A's\ are\ (B\&C)'s \end{array}$	$Q_3 = Q_1 \times Q_2$
rule (3) basic extension principle	$\begin{array}{l} Y = f(X_1, .., X_n) \\ X_i is A_i \quad i = 1, \ldots, n \\ \hline \text{Y is B} \end{array}$	$\mu_{f(A_1,..,A_n)}(v) =$ $sup_{u_1,\ldots,u_n}(\mu_{A_1}(u_1) \wedge \cdots \wedge \mu_{A_n}(u_n))$ $m_i = sup_u(\mu_A(u) \wedge \mu_{A_i}(u))$, $i = 1, \ldots, n$
rule (4) compositional rule of inference	$\begin{array}{l} X\ is\ A \\ (x, y)\ is\ B \\ \hline Y is A \circ B \end{array}$	$\mu_{A \circ B}(v) = sup_u(\mu_A(u), \mu_B(u, v))$
rule (5) Basic Probability	$\begin{array}{l} prob(XisA)\ is\ B) \\ \hline prob(XisC)\ is\ D \end{array}$	$\mu_D(v) = sup_r(\mu_B(\int_U(\mu_A(u)r(u)d(u)))$ $v = \int_U \mu_c(u)r(u)d(u), \quad \int_u r(u)d(u) = 1$ $r(u) = probability\ denssity\ function\ of\ u$

this framework is on using a probabilistic context-free grammar for translating the natural language sentences into GCs and protoforms.

None of the frameworks mentioned above presented a well-defined inference methodology that would be able to address the following issues:

– How to find the set of propositions in the knowledgebase that are relevant to the query?
– What is the inference chain for propagating constraints from a set of relevant propositions to the query?
– How to combine different answers obtained for the query?
– How to improve the quality of the answer obtained for the query?

This paper presents a reasoning methodology that addresses the above issues. The methodology that we propose here organizes the knowledge in a tree structure that we call a *constraint propagation tree* (CP). CPT extracts and organizes the set of relevant propositions in knowledgebase in response to a query. An evaluation algorithm then traverses the tree and propagates the constraints from this set to the query while aggregating different answers obtained for the query. CPT also allows one to identify the missing knowledge and establish a dialog with user when the information in knowledgebase is not enough for providing an answer.

2 The Reasoning Methodology

The reasoning methodology that is presented here takes the GC form of the query and the knowledgebase as input and makes a sequence of inferences to obtain a direct answer to the query. We assume the availability of a tool that translates the knowledgebase and the query in to generalized constraints.

The query posed to the system may be of various types. Generally a query can be viewed as seeking a value for one or more variables. Given the GC expression "*X is R?*", the query may ask for instantiation of the constraint variable (X) or the constraining relation (R). This view of the query includes a wide range of question types such as factual questions, list questions, definitions, and so forth. Our reasoning methodology instantiates the query variables in two phases: first the information relevant to the query is extracted and organized in a constraint propagation tree. Next the tree is evaluated to find the value for the query variables and combine different values obtained for these variables.

There are two types of relevancy: direct and indirect. Direct relevancy can be assessed by pattern matching while indirect relevancy requires reasoning and deduction on knowledgebase. For example if the query is Q: *"price(gas) is ?"*, and the knowledge base contains the propositions: p1: *"relation(price(gas), production(oil)) is direct"*, and p2: *"production(oil) is low"*, then p1 is directly and p2 is indirectly relevant to the query. Formally a proposition p is directly relevant to the query if it satisfies one the following conditions:

1. p contains the constraint variable and the subject of the query. For example p: *"relation(price(gas),production(oil)) is direct"*, is directly related to the Q: *"price(gas) is ?"*, because it contains the constraint variable of the query *price* as well as its subject *gas.*
2. p contains the constraint variable of the query with a generic subject. For example p: *"if Age(x) is young then risk(BreastCancer(x)) is high"*, is directly related to Q: *"risk(BreastCancer(Mary)) is ?".*

CPT applies the protoform deduction rules in a hierarchical way to extract the propositions that are directly or indirectly relevant to the query and determine how they are related. The root node in CPT represents the input query and the intermediate nodes are sub goals. Each node is connected to its children through a protoform rule, where the parent node represents the consequent and the children represent the antecedents of the rule. A node in CPT is represented by a tuple: (N, GC, E), where:

– N: is an integer that represents the node number.
– GC: is a generalized constraint that has zero or more uninstantiated variables, e.g., *"Age(Mary) is ?R"* or *"if Age(x) is over 40 then risk(breastCancer(x)) is high".*
– E: indicates the connection between the node and its children. $E = \{(r, \{C\})\}$, where r is the rule number and $\{C\}$ is a set of integers representing a group of immediate child nodes that form the antecedents of r. For example let us assume that a node i has children $\{j, k, m, n\}$ where nodes $\{j, k\}$ and $\{m, p\}$ are connected to their parent by rules a and b respectively. In this case $E = \{(a, \{j, k\}), (b, \{m, p\})\}$.

The following algorithm shows the procedure of generating a CPT.

```
Algoirthm CPTGeneration
Begin
  initialize the root node to the query
  repeat until no new nodes can be created
     let DRS be the set of propositions in KB that are directly
     related to the query.
     If DRS is not empty then
         for each proposition p in DRS
             if p matches with the query then
                instantiate the query variables
             if there is more than one instantiation for a variable then
                if the variable is veristic
                   instantiate it to the disjunction of individual values
              else instantiate it to the conjunction of individual values
             convert p and the query to protoforms: PF(p) and PF(Q)
             for each rule r in the protoform deduction rules:
                if PF(p) matches withthe antecedent of r &
                PF(Q) matches with the consequent of r then
                   create child nodes for the antecedents of r
         set the query to an uninstantiated leaf node
End
```

This algorithm first initializes the root node and extracts the set of propositions that are directly relevant to the query. This set is called directly related set (DRS). Then for each proposition in DRS, if it matches with the query, the query variables will be instantiated accordingly. This is the case where the answer to the query is explicitly stored in knowledgebase. For example, if we are interested to know the age of Mary: Q: *"Age(Mary) is ?R"*, and the knowledgebase contains the proposition *"Age(Mary) is middle-age"*, then we can instantiate R with the fuzzy subset that represents the granule value *"middle-age"*. If there is more than one proposition in DRS that matches with the query, the query variable will be instantiated to the conjunction of individual values. If for the above example KB also includes the proposition *"Age(Mary) is older than 30"*, then R will be instantiated to: *middle-aged* $\wedge$ *older than 30.* If the query variable is veristic [5], it will be instantiated to the disjunction of individual matches with DRS[1].

CPT allows seeking additional information from the user when the information in KB is not enough to answer a query. A leaf node in CPT can be tagged as a missing knowledge if it has at least one un-instantiated variable. Instantiating this variable may or may not be necessary for answering the query; however in the latter case it might improve the quality of answer by providing more constraints and thereby more robust estimates for the query variables.

[1] There are two classes of fuzzy variables verisitc and possibilistic. Possiblistic variables are disjunctive and can take only one value (e.g., *"Age(Mary)"*). In contrast the veristic variables are conjunctive and can take more than one value (e.g., *"big-Countries"*).

The second phase of reasoning is to propagate the constraints from the bottom of CPT to the top while combining different constraints obtained for each node. The propagation and aggregation algorithm is straightforward. It starts with the nodes in the level before the last level and applies the protoform inference rules (table 1) to the appropriate group of children to obtain a constraint for the parent node.If more than one value is obtained for a node variable then it will be instantiated to the conjunction of these constraints, however if the node variable is verisitic, it will be instantiated to the disjunction of the individual values.After instantiation, the value of a variable can be stored and reused for future queries, provided that the information about that variable will not change in the knowledgebase.It is worth noting that the fuzzy set obtained from applying the protoform rules should be normalized before being propagated to its upper level.

2.1 Evaluation of the Methodology

To evaluate our methodology, we applied it to a real world example taken from a web article about causes of breast cancer. Suppose that our knowledgebase consists of the following information:

The average chance that a woman being diagnosed by breast cancer is a function of age. From age 30 through age 39, it is about 0.4 %; from age 40 through age 49, it is about 1.5 %; from age 50 through age 59, it is about 2.5 % , and from age 60 through age 69 it is about 3.5 %. There are some factors that affect the average risk of breast cancer. Alcohol increases the average risk of breast cancer significantly; pregnancy in the age of 30 or before reduces the average risk of breast cancer by about 3 %, and in older women being overweight can increase the average risk of breast cancer slightly.

Suppose also that we have the following information about Mary:

Mary has a son who is about 20. She gave birth to her son when she was in her 20s. Mary is few years younger than Ann who is in her mid 50. Mary consumes about 1400 to 2000 calories a day. And she drinks moderately.

As rules of thumb, we also know that:

Overeating causes being overweight and the age of mother is equal to age of her son plus the age that she gave birth to her son.

Given the above information we are interested to know what Mary's chances of developing a breast cancer are. As mentioned before, the query and knowledgebase must be translated to GCs before the reasoning methodology can be applied. Generally this translation is not unique and depends on the question that is asked. Thus a proposition in knowledgebase can be translated according to all possible questions that may be asked about that proposition. Although this approach guarantees to find an answer for a question, if such answer exists, it can degrade the time performance considerably for large knowledgebases. A better approach is to find the questions that are most likely to be asked and translate the propositions in knowledgebase accordingly. For the purpose of this paper we assume that there exists a tool that performs such translation. By translating the above information to GCs we get:

1. *if age(x) is in 30s then average(risk(bc(x))) is about 0.4 % +If age(x) is in 40s then average(riskbc(x)) is about 1.5 % + If age(x) is in 50s then average(riskbc(x)) is about 2.5 % + If age(x) is in 60s then average(riskbc(x)) is about 3.5 %*

2. *if dirnkhabit(x) is regularly then alcoholFactor(riskbc(x)) is significant*
3. *if age(pregnancy(x)) is about 30 or before then pregnancyFactor(riskbc(x)) is about 3 %*
4. *if age(x) is old and weight(x) is overweight then weightFactor(riskbc(x)) is slightly*
5. *riskbc(x) is average(riskbc(x)) + alcoholFactor(riskbc(x)) + weightFactor(riskbc(x)) - pregnancyFactor(riskbc(x))*
6. *age(son(Mary)) is about 20*
7. *age(Mary) is Age(Ann) few years*
8. *age(Ann) is mid-50*
9. *age(pregnancy(Mary)) is in 20s*
10. *eatingHabit(Mary) is about 1400 to 2000 calories per day*
11. *if eatinghabit(x) is overeat then weight(x) is oveweight*
12. *age(x) = age(son(x)) +age(pregnancy(x))*
13. *drinkhabit(Mary) is moderate*

The CPT of this example is shown in figure 1. After defining appropriate fuzzy sets for the linguistic terms such as: *"about 3%", "old", "significant", "overweight", "mid-50", and so forth*, we calculated and deffuzified the answer as *"Risk(bc(Mary)) is 4 %"*.

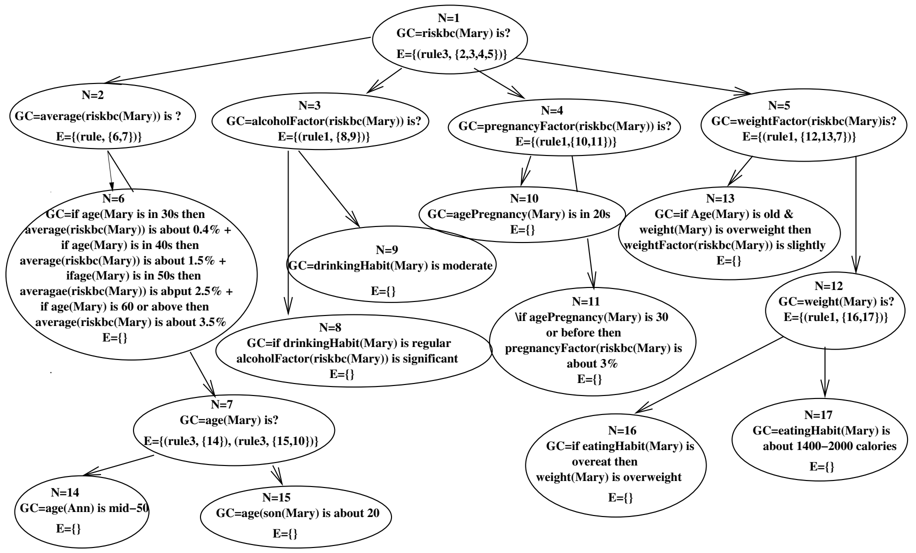
N=1
GC=riskbc(Mary) is?
E={(rule3, {2,3,4,5})}
N=2
GC=average(riskbc(Mary)) is ?
E={(rule, {6,7})}
N=3
GC=alcoholFactor(riskbc(Mary)) is?
E={(rule1, {8,9})}
N=4
GC=pregnancyFactor(riskbc(Mary)) is?
E={(rule1,{10,11})}
N=5
GC=weightFactor(riskbc(Mary)is?
E={(rule1, {12,13,7})}
N=6
GC=if age(Mary is in 30s then average(riskbc(Mary)) is about 0.4% + if age(Mary) is in 40s then average(riskbc(Mary)) is about 1.5% + ifage(Mary) is in 50s then averagae(riskbc(Mary)) is abput 2.5% + if age(Mary) is 60 or above then average(riskbc(Mary) is about 3.5%
E={}
N=10
GC=agePregnancy(Mary) is in 20s
E={}
N=13
GC=if Age(Mary) is old & weight(Mary) is overweight then weightFactor(riskbc(Mary)) is slightly
E={}
N=9
GC=drinkingHabit(Mary) is moderate
E={}
N=12
GC=weight(Mary) is?
E={(rule1, {16,17})}
N=11
\if agePregnancy(Mary) is 30 or before then pregnancyFactor(riskbc(Mary) is about 3%
E={}
N=8
GC=if drinkingHabit(Mary) is regular alcoholFactor(riskbc(Mary)) is significant
E={}
N=7
GC=age(Mary) is?
E={(rule3, {14}), (rule3, {15,10})}
N=16
GC=if eatingHabit(Mary) is overeat then weight(Mary) is overweight
E={}
N=17
GC=eatingHabit(Mary) is about 1400−2000 calories
E={}
N=14
GC=age(Ann) is mid−50
E={}
N=15
GC=age(son(Mary) is about 20
E={}

Fig. 1. The CPT of the example. Numbers of the rules are according to those listed in table 1.

3 Discussion and Summary

Current CW-based QA frameworks do not provide a systematic approach for extracting and combining the information in knowledgebase. In this work we developed a methodology that automates the process of inference in a CW-based QA system. The core of the methodology is the generation of a constraint propagation tree which extracts and organizes the knowledge relevant to the query. CPT also helps to achieve a more robust answer by identifying the missing information in knowledge base in response to a query.

Two issues remain to be addressed as a future work to scale up this methodology to a larger domain knowledgebase such as World Wide Web.

1. *Time performance.* An open domain QA system contains a vast dynamic knowledge source with various types of questions posed to the system. In such systems CPT can be excessively large and it may not be effective to generate and evaluate CPT for each question posed to the system. Thus appropriate techniques should be developed to store data from a previously generated CPT in an indexed database for use in later queries. This data should also be kept updated due to the highly dynamic nature of the web. In order to reduce the size of CPT, the generation algorithm can be modified to stop searching after finding a reasonable answer according to the user expectations.
2. *Commonsense knowledge.* The commonsense knowledge is usually generic, context dependent and uncertain (for example the famous proposition birds can fly). Including commonsense knowledge to the knowledgebase introduces nonmontonicity and adds a great complexity to the reasoning process. Dealing with commonsense knowledge is an open research area and is studied under the name of default reasoning [2,3].

References

1. Ahmad, R., Rahimi, S.: A Perception Based, Domain Specific Expert System for Question- Answering Support. In: WI 2006: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society Press, Los Alamitos (2006)
2. Dubois, D., Prade, H.: Approximate and commonsense reasoning: From theory to practice. In: Michalewicz, M., Raś, Z.W. (eds.) ISMIS 1996. LNCS, vol. 1079, pp. 19–33. Springer, Heidelberg (1996)
3. Schwartz, D.G.: Default reasoning with qualified syllogisms. In: ISUMA 1995: Proceedings of the 3rd International Symposium on Uncertainty Modelling and Analysis, pp. 396–401. IEEE Computer Society, Los Alamitos (1995)
4. Sufyan Beg, M.M., Thint, M., Qin, Z.: PNL-Enhanced Restricted Domain Question Answering System. In: Fuzzy Systems Conference, FUZZ-IEEE 2007, pp. 1–7 (2007)
5. Yager, R.R.: Veristic variables. IEEE Transactions on Systems, Man, and Cybernetics, Part B 30, 71–84 (2000)
6. Zadeh, L.A.: Precisiated natural language (PNL). AI Mag. 25, 74–91 (2004)
7. Zadeh, L.A.: Toward a generalized theory of uncertainty (GTU): an outline. Inf. Sci. Inf. Comput. Sci. 172, 1–40 (2005)